Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 4

По дисциплине «Основы алгоритмизации и программирования»

На тему «Представление информации в виде структуры»

Выполнил:

Студент 1 курса 10 группы

Жамойдо Артём Игоревич

Преподаватель: асс. Андронова М.В.

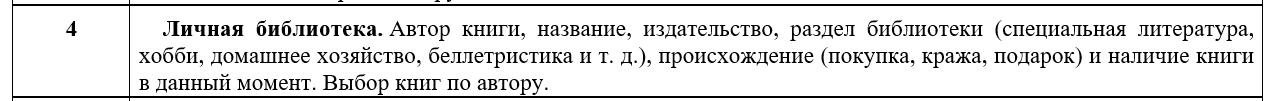
Минск, 2024

В соответствии со своим вариантом разработать программу для данных, приведенных в таблице ниже. Определить структурированный тип, разработать меню для работы с ***массивом структур***.

В программу должны войти функции:

* ввод элементов структуры с клавиатуры;
* вывод элементов структуры в консольное окно;
* удаление заданной структурированной переменной;
* поиск информации;
* запись информации в файл;
* чтение данных из файла.

**Вариант 4**



#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <Windows.h>

using namespace std;

// Прототипы

void addBook();

void displayBooks();

void deleteBook();

void searchByAuthor();

void saveToFile();

void readFromFile();

// Структура для представления информации о книге

struct Book

{

string author;

string title;

string publisher;

string category;

string origin;

bool available;

};

// Вектор для хранения информации о книгах в библиотеке

vector<Book> library;

// Функция для добавления книги в библиотеку

void addBook()

{

Book newBook;

// Ввод информации о книге

cout << "Введите автора: ";

getline(cin, newBook.author);

cout << "Введите название: ";

getline(cin, newBook.title);

cout << "Введите издателя: ";

getline(cin, newBook.publisher);

cout << "Введите категорию: ";

getline(cin, newBook.category);

cout << "Введите происхождение (покупка/кража/подарок): ";

getline(cin, newBook.origin);

cout << "Доступна ли книга в данный момент? (0 - Нет, 1 - Да): ";

cin >> newBook.available;

cin.ignore(); // Игнорируем символ новой строки

library.push\_back(newBook); // Добавляем новую книгу в библиотеку

}

// Функция для добавления книги в библиотеку

void displayBooks()

{

for (int i = 0; i < library.size(); ++i)

{

// Перебираем все книги в библиотеке и выводим информацию о каждой

cout << "Книга " << i + 1 << endl;

cout << "Автор: " << library[i].author << endl;

cout << "Название: " << library[i].title << endl;

cout << "Издатель: " << library[i].publisher << endl;

cout << "Категория: " << library[i].category << endl;

cout << "Происхождение: " << library[i].origin << endl;

cout << "Наличие: " << (library[i].available ? "Есть" : "Нет") << endl << endl;

}

}

// Функция для удаления выбранной книги из библиотеки

void deleteBook()

{

int index;

cout << "Введите индекс книги для удаления: ";

cin >> index;

// Удаляем книгу из библиотеки по указанному индексу

if (index >= 0 && index < library.size())

{

library.erase(library.begin() + index);

cout << "Книга успешно удалена" << endl;

}

else

{

cout << "Неверный индекс" << endl;

}

}

// Функция для поиска книги по автору

void searchByAuthor()

{

string searchAuthor;

cout << "Введите автора для поиска: ";

getline(cin, searchAuthor);

bool found = false;

// Ищем книги по заданному автору и выводим информацию

for (const Book& book : library)

{

if (book.author == searchAuthor)

{

cout << "Книга найдена:" << endl;

cout << "Автор: " << book.author << endl;

cout << "Название: " << book.title << endl;

cout << "Издатель: " << book.publisher << endl;

cout << "Категория: " << book.category << endl;

cout << "Происхождение: " << book.origin << endl;

cout << "Наличие: " << (book.available ? "Есть" : "Нет") << endl << endl;

found = true;

}

}

if (!found)

{

cout << "Книга автора " << searchAuthor << " не найдена" << endl;

}

}

// Функция для сохранения данных о книгах в файл

void saveToFile()

{

ofstream fout("library.txt");

if (!fout.is\_open())

{

cerr << "Ошибка открытия файла" << endl;

exit(1);

}

// Записываем данные о книгах в файл

for (const Book& book : library)

{

fout << book.author << endl;

fout << book.title << endl;

fout << book.publisher << endl;

fout << book.category << endl;

fout << book.origin << endl;

fout << book.available << endl;

}

fout.close();

cout << "Данные сохранены в библиотеку" << endl;

}

// Функция для чтения данных о книгах из файла

void readFromFile()

{

ifstream fin("library.txt");

if (!fin.is\_open())

{

cerr << "Ошибка открытия файла" << endl;

exit(1);

}

library.clear();

Book book;

// Читаем данные о книгах из файла

while (getline(fin, book.author))

{

getline(fin, book.title);

getline(fin, book.publisher);

getline(fin, book.category);

getline(fin, book.origin);

fin >> book.available;

fin.ignore(); // Игнорируем символ новой строки

library.push\_back(book);

}

fin.close();

cout << "Данные успешно прочитаны из библиотеки" << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

// Основное меню программы

do {

cout << "Меню:" << endl;

cout << "1. Добавление книги" << endl;

cout << "2. Вывод всех книг" << endl;

cout << "3. Удаление книги" << endl;

cout << "4. Поиск книг по автору" << endl;

cout << "5. Сохранение данных в библиотеку" << endl;

cout << "6. Чтение данных из библиотеки" << endl;

cout << "0. Выход" << endl;

cout << "Выбирите вариант: ";

cin >> choice;

cin.ignore(); // Игнорируем символ новой строки

// Обработка выбора пользователя

switch (choice)

{

case 1:

addBook();

break;

case 2:

displayBooks();

break;

case 3:

deleteBook();

break;

case 4:

searchByAuthor();

break;

case 5:

saveToFile();

break;

case 6:

readFromFile();

break;

case 0:

cout << "Завершение программы" << endl;

break;

default:

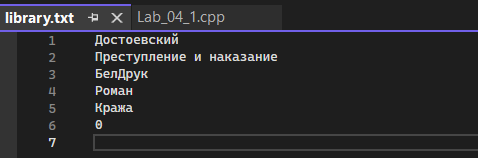
cout << "Неверный выбор. Попробуйте снова" << endl;

}

} while (choice != 0);

return 0;

}





Доп. задачи

1. Определить структуру для представления информации о сданных студентом экзаменах, содержащую поля: ФИО студента, число экзаменов, полученные оценки. Определить функции для обработки отдельного объекта (например, для проверки, сданы ли все экзамены на 4 и 5). Написать функцию для обработки массива структур. В результате обработки требуется вычислить характеристику успеваемости студентов, то есть отношение числа студентов, сдавших экзамены на 4 и 5, к общему числу студентов, в процентах.

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <Windows.h>

using namespace std;

// Структура для хранения результатов экзаменов студента

struct ExamResult

{

string fullName; // ФИО студента

int numExams; // Число экзаменов

vector<int> grades; // Полученные оценки

};

// Функция для ввода результатов экзаменов студента

void inputStudentExams(ExamResult& student)

{

cout << "Введите ФИО студента: ";

cin.ignore(); // Очистить буфер ввода

getline(cin, student.fullName); // Получаем ФИО студента

cout << "Введите количество экзаменов: ";

cin >> student.numExams;

cout << "Введите оценки за каждый экзамен:" << endl;

for (int i = 0; i < student.numExams; ++i)

{

int grade;

// Получаем оценку и добавляем в вектор оценок

cout << "Экзамен " << i + 1 << ": ";

cin >> grade;

student.grades.push\_back(grade);

}

}

// Функция для вывода результатов экзаменов студента

void displayStudent(const ExamResult& student)

{

cout << "Студент: " << student.fullName << endl;

cout << "Количество экзаменов: " << student.numExams << endl;

cout << "Оценки: ";

for (int grade : student.grades)

{

cout << grade << " ";

}

cout << endl;

}

// Функция для вычисления процента студентов, сдавших экзамены на 4 и 5

float calculatePerformance(const vector<ExamResult>& students)

{

int passedCount = 0;

for (const ExamResult& student : students)

{

bool allPassed = true;

for (int grade : student.grades)

{

if (grade < 4)

{

allPassed = false; // Если есть оценка ниже 4, студент не сдал

break;

}

}

if (allPassed)

{

passedCount++;

}

}

// Возвращаем процент сдавших

return static\_cast<float>(passedCount) / students.size() \* 100;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int numStudents;

cout << "Введите количество студентов: ";

cin >> numStudents;

cin.ignore(); // Игнорируем символ новой строки

vector<ExamResult> students(numStudents); // Создаем вектор студентов

for (int i = 0; i < numStudents; ++i)

{

cout << "Введите данные студента " << i + 1 << ":" << endl;

inputStudentExams(students[i]);

}

cout << endl << "Информация об экзаменах студентов: " << endl;

for (const ExamResult& student : students)

{

displayStudent(student);

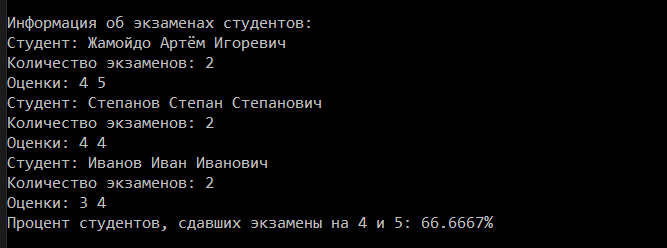
}

float performance = calculatePerformance(students);

cout << "Процент студентов, сдавших экзамены на 4 и 5: " << performance << "%" << endl;

return 0;

}



2. Описать структуру с именем TRAIN, содержащую поля: названия пункта назначения, номер поезда, время отправления. Написать программу, выполняющую ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN (записи должны быть размещены в алфавитном порядке по названиям пунктов назначения); вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких поездов нет, то вывести сообщение об этом).

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <Windows.h>

using namespace std;

// Структура для хранения информации о поезде

struct Train

{

string destination; // Пункт назначения

int trainNumber; // Номер поезда

int departureTime; // Время отправления

};

// Функция для сравнения поездов по названиям пунктов назначения

bool compareTrains(const Train& train1, const Train& train2)

{

return train1.destination < train2.destination;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int numTrains = 8;

vector<Train> trains(numTrains); // Создаем вектор поездов

// Ввод информации о поездах

for (int i = 0; i < numTrains; ++i)

{

cout << "Введите информацию о поезде " << i + 1 << ":" << endl;

cout << "Пункт назначения: ";

cin >> trains[i].destination;

cout << "Номер поезда: ";

cin >> trains[i].trainNumber;

cout << "Время отправления: ";

cin >> trains[i].departureTime;

}

// Сортируем поезда по названиям пунктов назначения

sort(trains.begin(), trains.end(), compareTrains);

int time;

cout << "Введите текущее время: ";

cin >> time;

bool foundTrain = false;

cout << "Поезда, отправляющиеся после " << time << ":" << endl;

// Поиск и вывод информации о поездах, отправляющихся после указанного времени

for (const Train& train : trains)

{

if (train.departureTime > time)

{

foundTrain = true;

cout << "Номер поезда: " << train.trainNumber

<< ", Пункт назначения: " << train.destination

<< ", Время отправления: " << train.departureTime << endl;

}

}

if (!foundTrain)

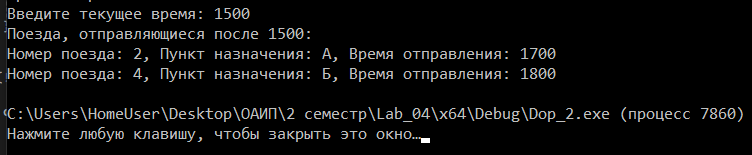
{

cout << "Поездов, отправляющихся после указанного времени, нет." << endl;

}

return 0;

}



3. Определить структуру для представления информации о наличии санаторных путевок, содержащую следующие поля: Название санатория, Место расположения, Лечебный профиль, Количество путевок. Представить введенные данные в виде таблицы, сгруппировав их по лечебным профилям санаториев. В пределах каждой группы данные отсортировать по названиям санаториев. Организовать поиск информации и вывод результатов.

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <Windows.h>

using namespace std;

// Структура для хранения информации о санаторных путевках

struct Sanatorium

{

string name;

string location;

string profile;

int availability;

};

// Функция для сравнения санаториев по названию

bool compareSanatorium(const Sanatorium& s1, const Sanatorium& s2)

{

return s1.name < s2.name;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int numSanatoriums = 4;

// Создание вектора структур Sanatorium

vector<Sanatorium> sanatoriums(numSanatoriums);

// Ввод информации о санаторных путевках

for (int i = 0; i < numSanatoriums; ++i)

{

cout << "Введите информацию о санатории " << i + 1 << ":" << endl;

cout << "Название санатория: ";

cin >> sanatoriums[i].name;

cout << "Местоположение: ";

cin >> sanatoriums[i].location;

cout << "Лечебный профиль: ";

cin >> sanatoriums[i].profile;

cout << "Количество путевок: ";

cin >> sanatoriums[i].availability;

}

// Сортировка санаториев по лечебному профилю

sort(sanatoriums.begin(), sanatoriums.end(),

[](const Sanatorium& s1, const Sanatorium& s2)

{

return s1.profile < s2.profile;

});

// Вывод информации в виде таблицы, сгруппированной по лечебным профилям и отсортированной по названиям санаториев

cout << "----------------------------------------------" << endl;

cout << "Название санатория | Местоположение | Кол-во путевок" << endl;

cout << "----------------------------------------------" << endl;

string currentProfile = "";

for (const Sanatorium& sanatorium : sanatoriums)

{

if (sanatorium.profile != currentProfile)

{

cout << "Лечебный профиль: " << sanatorium.profile << endl;

currentProfile = sanatorium.profile;

}

cout << sanatorium.name << " | " << sanatorium.location << " | " << sanatorium.availability << endl;

}

// Поиск информации

string searchProfile;

cout << "Введите лечебный профиль для поиска: ";

cin >> searchProfile;

cout << "Результаты поиска для лечебного профиля '" << searchProfile << "':" << endl;

for (const Sanatorium& sanatorium : sanatoriums)

{

if (sanatorium.profile == searchProfile)

{

cout << sanatorium.name << " | " << sanatorium.location << " | " << sanatorium.availability << endl;

}

}

return 0;

}

